A starter guide for model evaluations

This is a starter guide for model evaluations (evals). Our goal is to provide a general overview of what evals are, what skills are helpful for evaluators, potential career trajectories, and possible ways to start in the field of evals.

Evals is a nascent field, so many of the following recommendations might change quickly and should be seen as our current best guess.

Why work on evals?

Model evaluations increase our knowledge about the capabilities, tendencies, and flaws of AI systems. Evals inform the public, AI organizations, lawmakers, and others and thereby improve their decision-making. However, similar to testing in a pandemic or pen-testing in cybersecurity, evals are not sufficient, i.e. they don't increase the safety of the model on their own but are needed for good decision-making and can inform other safety approaches. For example, evals underpin Responsible Scaling Policies and thus already influence relevant high-stakes decisions about the deployment of frontier AI systems. Thus, evals are a highly impactful way to improve the decision-making about AI systems.

Evals are a nascent field and there are many fundamental techniques to be developed and questions to be answered. Since evals do not require as much background knowledge as many other fields, it is much easier to get started and possible to make meaningful contributions from very early on.

What are model evaluations (evals)?

Evals refers to a broad category of approaches that we roughly summarize as:

The systematic measurement of properties in AI systems

More concretely, evals typically attempt to make a quantitative or qualitative statement about the capabilities or propensities of an AI system. For example, we could ask if a model has the capability to solve a specific coding problem or the propensity to be power-seeking. In general, evals are not restricted to safety-related properties but often when people talk about evals, they mention them in a safety context.

There is a difference between *red-teaming* and *benchmarking*[1]. Red-teaming is actively looking for specific capabilities or propensities while interacting with the model. It is an attempt to answer the question "Can we find this capability in a model when we try hard to find it?". In other words, red-teaming is an attempt to show the existence of certain capabilities/properties, but it is

[1] both of which are evals

not trying to make a claim about how likely those are to occur under real-use conditions. Red-teaming typically involves interacting with the model[2] and actively looking for ways to elicit the desired behavior, e.g. by testing many different model inputs and strategies and actively iterating on them.

In contrast, benchmarking makes a statement about the *likelihood* of a model behaving in a specific way on a certain dataset, e.g. the likelihood of a behavior occurring under real-use conditions. A benchmarking effort should be designed while interacting with the model as little as possible in order to prevent overfitting to the capabilities or tendencies of any particular model.

Both red-teaming and benchmarking are important and serve a purpose. Red-teaming can provide an estimate of the potential danger of a system, e.g. whether the model can manipulate its users. Benchmarking can provide an estimate of how likely an AI system would show these tendencies under specific conditions, e.g. how likely the model is to manipulate its users in realistic scenarios. Currently, evals are often a mix between red-teaming and benchmarking but we expect the two categories to get more and more distinct.

There is a difference between capability and alignment evaluations. Capability evaluations measure whether the model has the *capacity* for specific behavior (i.e. whether the model "can" do it) and alignment evaluations measure whether the model has the *tendency/propensity* to show specific behavior (i.e. whether the model "wants" to do it). Capability and alignment evals have different implications. For example, a very powerful model might be capable of creating new viral pandemics but aligned enough to never do it in practice[3].

Currently, evals are mostly associated with behavioral measurements but they could also include interpretability/explainability tools. For example, once the technical tools are available, there could be a range of interpretability-based evals that would provide more detailed information than behavioral tests alone.

Behavioral evals have clear limitations and it's important to keep that in mind. Every behavioral test will always be spotty and cover a small slice of the potential input space. While we can use behavioral evals to get hints at what the internal mechanisms within an AI system might be, similar to how psychology can make statements about the internal mechanisms of the brain, it is by no means as precise as good interpretability tools would be. Thus, we think of evals as a way to reduce uncertainty from very uncertain to less uncertain but to make high-confidence statements, we should not rely on evals alone.

[2] This interaction can also happen automatically, e.g. in the case of automated red-teaming
[3] This also applies to the misuse case where e.g. the model has the knowledge of how to build a bomb but is aligned enough to never reveal that knowledge to a user.

What skills are helpful for evaluators?

The following is a list of qualities that we think are generally helpful for evaluators. Note, that they are by no means necessary, i.e. you can meaningfully contribute without having mastered these skills. Our suggestions should be seen as pointers to helpful skills rather than requirements.

LLM steering

By LLM steering[4], we mean the ability to get an LLM to do specific things. In this case, LLM serves as a placeholder for whatever the state of the art in AI systems is. Since model evaluators typically make statements about the maximum capacity of a model, working with state-of-the-art systems is required. Currently, these are language-based models but frontier models are already increasingly multi-modal. Thus, the list of suggestions below should be extended with whatever skills are required to steer state-of-the-art models and elicit their properties.

Prompting

The most obvious form of getting a model to do what we want is by prompting it in clever ways.

Thus, evaluators should **know the basics of prompt design**. This can include knowing a particular set of prompts that works well for a given model, knowing basic prompt pieces that can be put together to form a more efficient prompt, or even better, having a more general predictive theory of how “the model works”.

In some cases, we want to elicit properties of models that have already been fine-tuned to not show these qualities, e.g. by RLHF. Therefore, the ability to **break a model or do prompt injections** seems really helpful to a) show the limitations of such fine-tuning attempts and b) get the model to elicit quantities that the model doesn’t show with standard prompting techniques.

To get started, you can check out a prompting guide by Anthropic, Hugging Face, or PromptingGuide.

Playing with LLMs

In our experience, getting a “feeling for the model” is very important. This means refining your intuition for how models would typically react to many different prompts, which type of things they are good or bad at, what different strategies can be used to make them output certain texts, etc. Often, we found it hard to formalize this knowledge or transfer it between people with different levels of experience. A lot of this informal knowledge comes from “playing around” with the model, interacting with it, trying to jailbreak it, and applying new discoveries yourself (e.g. Chain of thought, Learning from Language Feedback, LM agents, etc.). While playing with the

[4] Not to be confused with activation steering

model, you often stumble upon something curious, quickly form a hypothesis, and check it with a few additional examples. This is much more uncertain than rigorous scientific research but sharpens and refines your intuitions a lot which you can then use in your scientific endeavors.

Supervised fine-tuning (SFT)

If a malicious actor wanted to get a model to act in specific (bad) ways, they would likely fine-tune the model rather than just prompting it. The same is true for a misaligned model that wanted to self-improve for nefarious purposes. Therefore, model evaluators should conduct gain-of-function research in controlled and safe environments to elicit these behaviors. Fine-tuning can also be helpful in other ways, e.g. to test how easy it is to undo the guardrails of the model or fine-tune a specialized helper model for a more complex task.

Thus, it is helpful to **know how to finetune LLMs** (both with API finetuning and open-source model finetuning). Relevant skills include how to use GPUs and parallelize your fine-tuning jobs, and implicit knowledge about batch sizes, learning rates, optimizers, quantization, data augmentation, and more.

To get started, you can check out guides by Lakera, Maya Akim, Hugging Face, or the OpenAI finetuning API.

RL with LLMs

In some cases, we should aim to use RL-based finetuning to elicit a particular behavior. Especially in cases where the model is supposed to show agentic behavior, RL-based finetuning seems preferable over SFT.

Therefore, it is helpful to be familiar with the combination of RL and the type of model you're evaluating, e.g. LLMs. Useful knowledge includes how to set up the pipeline to build well-working reward models for LLMs or how to do "fake RL" that skips training a reward model and replaces it e.g. with a well-prompted LLM.

To get started, you can check out tutorials from Hugging Face, Weights and Biases, and Labellerr.

Scaffolding and LM agents

Often, we want to understand the behavior of LLMs in more complex settings than just question-answering. For example, the LLM might be turned into an "LM agent" through scaffolding, i.e. we build software and tools around the LLM that allow it to continuously act in a real or simulated environment more naturally than just an LLM alone would.

Scaffolding is a nascent field, so there is no well-established definition or methodology but it's helpful to be good at prompt engineering and building software frameworks around your model.

We expect that LM agents will become very prominent soon, so we especially recommend trying to understand them in detail.

To get started, you can check out a blog post by Lilian Weng and METR's paper.

Tool use

There are a couple of tools that might be helpful to know for evals, e.g. Langchain or the OpenAI evals framework. Both of these are early-stage packages and come with their own flaws but they might still be helpful in your workflow or inspire you to write better ones.

Generalist

In our experience, most evals projects iterate between two parts.

1. **Conceptual:** This can include answering questions like: what property am I trying to measure? What is the best way of measuring it? What experiments would give me strong evidence about that property? And more.
2. **Execution:** This can include coding up the experiments, running the evals on different models, evaluating the results, writing up the findings, and more.

Given the early state of the field of evals, there are a lot of straightforward questions where the conceptual part is not very complicated and most of the project depends on execution. In these cases, most of the progress comes from getting simple or medium complex tasks done quickly and less from particularly deep/formal insights.

Therefore, evaluators benefit from a **generalist skillset** that allows them to draw from a range of experiences and a "**get stuff done**" **attitude** that enables them to execute the low-hanging fruit experiments quickly. For example, an evaluator benefits from being willing to learn new LLM-related techniques quickly and applying them hands-on since the field is moving so quickly.

Nevertheless, the conceptual part should not be neglected. Potentially, the most relevant safety-critical evidence could come from a small number of well-designed experiments that ask exactly the right question. Those experiments can be the result of multiple weeks or months of thinking about conceptual questions or tinkering with different setups before jumping to the execution phase. Thus, the more mature the field of evals becomes, the more specialized its members will be and the more they benefit from specialized skills rather than a generalist skillset. However, in the current state of the field, a generalist skillset is very helpful. From our own experience, we expect that hands-on experience will always be extremely valuable for evals and thus recommend against specializing entirely in conceptual work.

Scientific mindset

By default, most model evaluations will have multiple potential interpretations. Thus, a beneficial skill for model evaluators is having a scientific mindset. Concretely, this means keeping

alternative explanations for the results in mind and tracking potential confounders. Optimally, these plausible alternative hypotheses are then used to identify, design, and run experiments to test the potential confounders and identify the true effect.

Evals are especially prone to Goodharting, i.e. someone designs a benchmark for a specific target quantity, then people use that benchmark as the sole measure for the target at which point it ceases to be a good benchmark. Thus, a good model evaluator should aim to **red-team** the current suite of benchmarks and look for ways in which they are measuring the wrong proxy. Optimally, they are then able to design a wide variety of experiments that cover many different angles and are sufficiently redundant to further decrease the probability of misinterpretation.

Empirical research experience

In practice, it is non-trivial to get this scientific mindset but in our experience, you learn how to science by doing science. Concretely, this means doing research projects with a more experienced supervisor and working on scientific projects. For example,

- You can work on scientific projects in your Bachelor's or Master's degree. Often professors look for research assistant positions. If you find something you're broadly interested in, a research assistant position can be very helpful. At this stage, it likely doesn't matter whether the research is closely related to evals, as long as it's broadly in the field of ML.
- You can take thesis projects seriously and aim for a (small) publication. If you're willing to put in the effort most supervisors are likely to support you in your attempt to publish your thesis as a workshop paper or a conference publication. This typically requires you to put in significantly more effort than the thesis itself but it teaches you a lot about scientific practices and writing.
- You can do programs like MATS outside of university. Often there are great benefits from being in an environment where many people are working on related projects and can discuss their experiences and findings with each other.
- Potentially it is worth attempting a PhD. While people disagree about whether a PhD is necessary to be a good scientist, it is true that most people are much better scientists after doing a PhD.
- It is possible to develop a scientific mindset with little or no supervision. We suggest starting with a project that is well-scoped and simple, e.g. reproduce or extend existing work. Since evals is a nascent field, many simple questions can be attempted with little or no supervision.

Especially valuable is research experience that involves different kinds of LLM steering, e.g. prompting, fine-tuning, RL with LMs, or LM agents.

Software engineering

Many tasks in evals benefit from a **solid software engineering background**. This can include designing scaffolding around the model, building APIs around various tasks, basic data science, basic database management, basic GUI design, and more.

While there is some helpful theoretical knowledge to be learned about software engineering (see e.g. A Philosophy of Software Design or Clean Code), we expect most benefits to come from practical experience. Fortunately, with modern LLMs, learning software engineering has become easier than ever, both because you can iterate faster and because you can use LLMs to provide feedback on your code and suggest general improvements.

Potential career paths

Most technical paths are not static. Some people who start in engineering become more and more focused on research over time and others who start as scientists focus more on engineering later. Thus, deciding in favor of one of these paths today does not mean you can't switch in the future and most skilled model evaluators are decent at both.

Please note that it's not at all necessary to be good at all of these skills before you can contribute. Our suggestions merely serve as a pointer for which skills are especially helpful.

Engineering-focused

The engineering spectrum can range from pure software engineering to research engineering. On the pure software side of the spectrum, the engineer would be mostly building and improving tools for the scientists in the team, and on the research side of the spectrum, they would also help design and set up experiments.

- **LLM steering:** The core competency of an evals engineer is LLM steering. Depending on the project, this can be more focused on prompting, finetuning, or scaffolding but you're almost certainly going to need all three of these skills at some point.
- **API building & usage:** The ability to design and build APIs for your scaffolding or evals and to understand how APIs from various providers work and how they can be integrated into the current evals stack.
- **Basic data analysis & management:** A large part of evals is the creation, curation, and efficient management (i.e. storage and pipelining) of data. Thus, being able to use data management systems and basic data analysis tools comes in handy.
- **Basics of experimental design:** identify the target quantity, and design a setup that measures the target quantity while reducing the chance of measuring other proxy variables.
- **Plotting:** this is just a basic tool that everyone working in research should have.
- **UI and tooling building:** design and implement tools for technical and non-technical users, for example, a CLI tool with OpenAI API-like experience for finetuning custom

models on internal compute, or setting up and maintaining an OpenAI Playground-like UI for quick model testing.
- **DevOps and Infrastructure:** provisioning and managing compute resources for researchers.

Research-focused

Evals research scientists are likely more hands-on compared to other research scientist positions. Helpful skills include:

- Competent **research engineering** is almost certainly necessary to be a good evals research scientist. Thus, the qualities of the research engineer apply. However, the focus on pure software design is lower than for a research engineer.
- **Experimental design:** Similar to research engineering but with a stronger focus. Experience in scientific research is helpful for this skill.
- **Conceptual research:** Beyond experimental design, it is of similar importance for a research scientist to have conceptual clarity and a high-level understanding about which experiments matter for which reasons, e.g. by having a high-level roadmap or a concrete agenda for their research.
- **Writing:** Experimental results have to be communicated within the team, with external collaborators, and often with the wider public or regulators. Therefore, writing and communication skills are key for a research scientist position.

How to start with evals work?

There are many different ways to get started. Here, we suggest two common approaches.

Hands-on approach

One possible way to start with evals is to “Just measure something and iterate”. The broad recipe for this would be:

1. Pick a quantity you find generally interesting and would want to understand more deeply.
2. Play around with the model (e.g. in the OpenAI playground) to see if you can find simple and unprincipled ways to measure the behavior.
3. Abstract and formalize your testing procedure and evaluate the model more rigorously.
4. Identify the weaknesses and limitations of your current way of measuring.
5. Refine and extend your evaluations.
6. Iterate until you have a sound and usable evaluation.

This approach may be the right choice for you if you prefer to work on concrete projects rather than learning general skills, like to learn things on the fly and enjoy iterating on empirical feedback.

This approach is more likely to be successful if you have some mentorship. While it may be hard to get mentorship as a newcomer to the field, often researchers are responsive to a high-quality research proposal. Nevertheless, we want to emphasize that it is entirely possible to do good work entirely without mentorship and we can wholeheartedly recommend just giving it a go.

Learning general skills approach

Rather than working on a concrete project, you can also try to improve your basic evals skills in general similar to how students learn core skills in University and only apply them later.

This could, for example, entail going through a general prompting tutorial, fine-tuning different models in a supervised or RL-based fashion, building an LLM agent with scaffolding, and playing around with the existing tools that are helpful for evals. In this case, you would do all of this without having any specific evaluation in mind.

This approach may be the right choice for you if you're very new to software or ML engineering or you have a general preference for learning basic skills before applying them.

Marius' opinion: While there is no clearly correct way to get into evals work and it depends on personal preferences and the level of skills, I strongly recommend attempting a hands-on project. I think this is the fastest way to get good at evals, gives good evidence about where your skills are and whether you enjoy the process. If you realize that you lack some of the core skills, you can still switch to learning general skills. We made the fastest progress by interacting with LLMs a lot and I generally recommend this approach even if you're fairly new to LLMs.

Contributions

Marius Hobbhahn led this post, drafted the first version, and edited the final version. Mikita Balesni, Jérémy Scheurer, Rusheb Shah, and Alex Meinke gave feedback.

We shared this post with participants of the Apart Research Evals Hackathon on 24 November 2023 and are thankful for feedback from participants.